SPEAKERPHONE : REAL-TIME DEVICE EMULATION

Jesca Rachelle Chengula, Beau Hanson, Aaron Messina

University of Rochester, 2020

ABSTRACT

The goal of this project is to create a library of low-quality speaker and device emulations. Using a collection of algorithms to define the features of a device, a user is then able to input any audio signal through the modeled device offline, or in real-time. The three most salient features of the device modeling are filtering, distortion, and compression. As such, the desired device is characterized with feature sets for each of these three parameters, optimized for real-time processing, then processed with each algorithm. With this program, a user can hear anything input processed through the device of their choosing.

Index Terms— filtering, compression, distortion, degradation

1. INTRODUCTION

To implement the emulation of some of Speakerphone's features on our own, experimentation with several devices of varying quality had to be conducted to learn how a given signal is affected as it is outputted through each device. Some of our chosen devices included iPhones, Samsung Galaxy S9 and the Subaru speaker system. The devices' output quality was tested using a few different audio recordings, including a white noise signal, a song and a sine sweep. Test files of 16-bit depth and in 44.1 kHz, and songs with a wide variety of frequency content were used to encourage the best results. The recorded test files are outputted through each device and the data read into MATLAB and is compared with the raw data (i.e. the unaltered test recordings). The comparisons were made after performing Fourier Transform to each signal to better understand the frequency content of each.

2. FILTERING

The process of filtering the audio signal needs to be efficiently defined for ease of real-time implementation. For this reason, an FIR filter was algorithmically chosen to fit the frequency response of the input device. The algorithm to define the filter is not, however, inhibited by a low-latency requirement. The fitting algorithm is thus a thorough process to initialize the FIR filter given ranges of values for cutoff frequencies and filter order.

2.1. Frequency response of the device

To easily analyze the frequency response of a device, it is necessary to have some sort of equal power input. White noise was chosen as it enabled the chosen length to be easily manipulated as opposed to something like an impulse. Once white noise was recorded through the device, it can be directly compared with the unaffected white noise input signal. The simple filter equation below demonstrates how to retrieve the filter from a processed and unprocessed signal.

$$H(j\omega) = \frac{Y(j\omega)}{X(j\omega)}$$

This operation can be quickly calculated for each device as a first step to defining the fitted filter.
The final step to deriving this filter is to smooth and normalize the data such that it can be interpreted easily by a simple bandpass filter. Any insignificant peaks in the pass band are smoothed to create a flatter pass band. The normalization occurs in the linear scale, and then the filter is translated to dB scale.

2.2. Filter fitting algorithm

Once the derived filter is normalized and smoothed, the program finds a multi-order FIR filter of best fit. There are three variables associated with the filter: low frequency cutoff, high frequency cutoff, and filter order. Given that the filter needs to have great flexibility to accurately characterize the input device's frequency response, these variables cover a wide range. The three variables are used in every permutation, each one creating a test filter. A calculation of mean square error (see below) then compares

the test filter with the device-derived filter. A simple minimum-value comparison of all of the permutations' mean square errors chooses the filter with the best fit, and then saves the filter coefficients. The filter coefficients can then be used in real-time with a simple "Direct Form II Transposed" filtering algorithm.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

2.3. Filter evaluation

As a method of evaluation for the filter, one can simply evaluate the mean square error already calculated in the algorithm, and determine if it satisfies a set threshold. If the value falls beneath the threshold, it is said to be an appropriate filter, but if the MSE value exceeds the threshold, a more broad filter can be implemented to then narrow down the appropriate values for cutoff frequencies and filter order. There is also a consideration to make for aural similarity. For instance, the fit filter with the lowest MSE value may not be the one that aurally sounds the most similar to the original source. In that case, it is possible to simply adjust the frequencies or filter order to better accommodate the aural similarities. See below an example of a fit filter for the iPhone 5 model. The mean-square-error for this filter was approximately 2000 with an acceptable threshold of 3000.

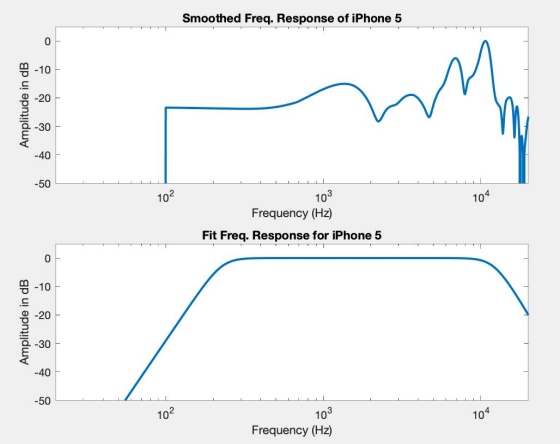

Figure 1: Frequency response of an iPhone 5, and the fit filter that was algorithmically derived.

3. COMPRESSION

Compression is one among the methods of controlling the dynamic range of a signal. Compression reduces the dynamic range of the signal in which case the loud bits are attenuated and the quiet bits are boosted. The make up gain is used to boost the compressed signal as the act of compressing it leads to a significant attenuation of the signal. To set up a compressor, controls such as threshold, ratio, attack, release, knee, make-up gain and output are set. the compression ratio is given by the formula below;

$$R = \frac{XdB - CT}{YdB - CT}$$

R>0 for compression to act on the signal. Compression in our speakerphone is used to emulate and enhance the guitar input signal and other recordings.

3.1. Using the compression function

To apply compression into the uncompressed signal, the compression function was adapted from 2002 DAFx book by Zolzer. The input parameters taken in are the uncompressed signal(x), release, attack, the compression parameter '**comp**', the filter parameter and the sampling rate of the signal. The attack determines how quickly the compressor starts to work. Release determines how soon after the signal dips below the threshold the compressor stops. The compression parameter shows how much compression is applied based on 0>comp>-1. This translates that, the lower the comp value, lesser compression is applied while a higher comp value results into a more compression. Upon passing the uncompressed signal into the compression function, the compression signal is multiplied by the gain reduction. The gain reduction regulates the amplitude of the signal to prevent it from becoming too loud or keeping the signal at a relatively uniform level. The figure below illustrates the difference between the compressed and the uncompressed signal of Peg, a song by Steely Dan with sampling rate of 44.1kHz and bit depth of 16.

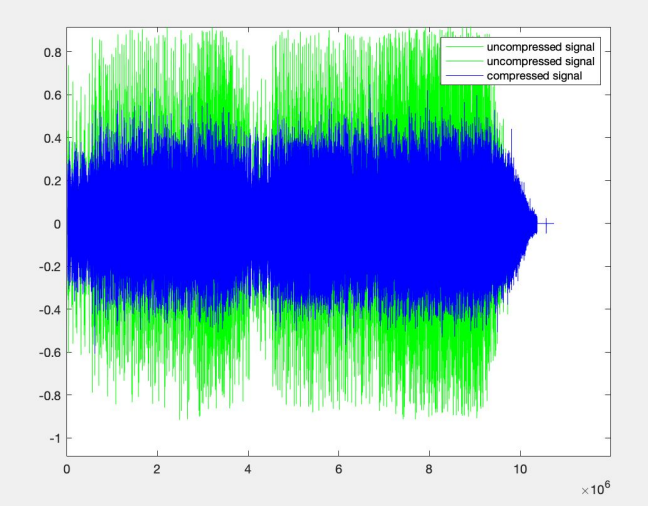

Figure 2: Peg song signal before and after compression is applied

4. DISTORTION

Many of the low quality speakers emulated in this project exhibit signs of harmonic distortion. While some of the devices produce effects of distortion that are nearly imperceptible, some, such as the Subaru Forester speaker system, introduce a fair amount. Therefore, in order to pursue the common goal of an accurate representation for *every* device in the library, the presence of a harmonic distortion algorithm with manipulatable intensity is necessary to include in the overall process.

4.1. Harmonic distortion theory

Due to their simplicity, sine waves are ideal candidates for basic signal analysis. An ideal sine wave contains only one frequency, often denoted f1. Figure (3a) shows the magnitude spectrum of this theoretical signal. A nonlinear sine wave that has encountered distortion, however, may have a magnitude spectrum similar to that of Figure (3b). It

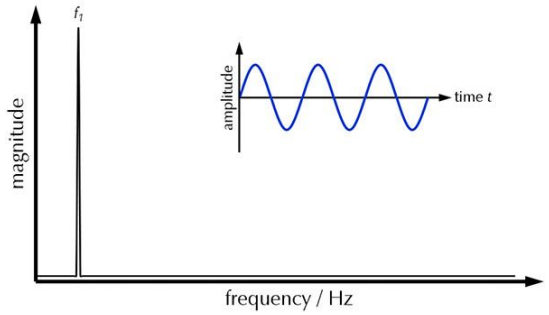

Figure 3a: FFT spectrum of an ideal sine wave [3]

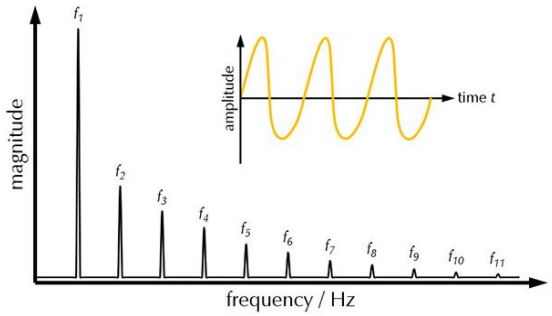

Figure 3b: FFT spectrum of a distorted sine wave [3]

can be observed that this spectrum contains a number of different frequencies higher than f1. The presence of these harmonics is a distinctive sign of harmonic distortion, and their frequencies are integer multiples of f1. In electronics, a circuit containing an input signal with a voltage greater than the maximum voltage the power supply can provide can lead to a 'clipped' output voltage. Some of the devices in the library have the potential to output such waveforms. In signal processing, this effect can be modeled by amplifying only the *odd* harmonics to the input waveform. In an extreme case, this process would approximate a square wave output, assuming the input is an ideal sine.

4.2. Distortion algorithm

The distortion algorithm used consists of two stages: fundamental pitch identification and parametric filter design and application. These stages are performed in a single function that takes in an input frame of audio, its sample rate and two other customizable parameters that will be described shortly. The first stage analyses the FFT of the input frame to find its maximum frequency, which is assumed to be the fundamental frequency constituting the frame. The second stage generates a multi-band parametric filter designed to emphasize the desired harmonics. This filter takes in, as an argument, the center frequency of each band. These frequencies are calculated simply as odd integer multiples of the assumed f1 (i.e. the odd harmonics). The filter also takes in the gain of each band, as well as the bandwidth. Both of these quantities are inversely related to the frequency, ensuring that the harmonics become less prevalent as frequency is increased. Following the generation of the filter, the input frame is processed through it, resulting in an output with additional harmonic distortion. As a simple demonstration, a 1kHz sine wave is passed through the function, and the resulting magnitude spectra are shown in Figure (4a) and (4b).

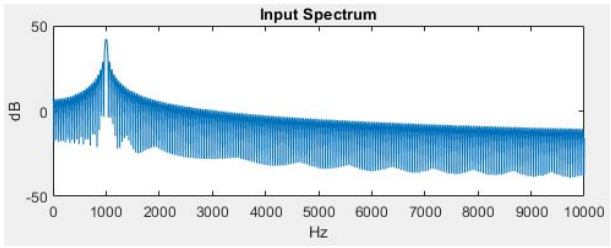

Figure 4a: 1kHz 'pure' sine wave magnitude spectrum

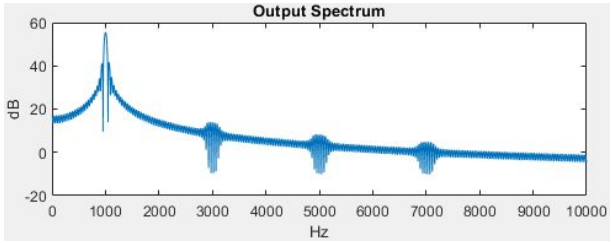

Figure 4b: 1kHz sine wave with harmonic distortion

4.3. Fitting distortion to device

Since every device in the library introduces a different level of distortion, it was necessary to define some adjustable parameters to fit the amount of distortion to each device's speaker. These include the number of amplified harmonics and the intensity (gain) of the parametric filter. Both of these quantities have a strong effect on the overall processing of a given input. The intensity, which for simplicity is scaled to range from 0 to 1, directly affects the gain of the first and most prevalent band in the filter. The gain of the other bands are based off of this 'initial' band and they fall off exponentially with increasing frequency. The values were chosen for each device based off of a number of tests. Many pure sine tones, as well as some music samples, were passed through the function to listen and compare the output waveform to those outputted directly from the device of interest. This process of trial and error resulted in a list of values containing the key features for each speaker's harmonic distortion.

5. REAL -TIME IMPLEMENTATION

The real-time implementation of the device emulation is a relatively simple process in three steps. First, a device is selected through which one wishes to process audio. The selection of the device includes fit IIR filter coefficients, compression coefficients for ratio, attack, release, knee, and make-up gain, and finally the distortion parameters: number of harmonics and level of harmonics. Given that all of these parameters are already defined with the previous algorithms, one can simply process audio by buffer through each of the three processes, and hear the audio output as if it were being played through the selected device.

6. CONCLUSION

After developing a small collection of device models by method of filtering, distortion and volume control, it can be concluded that breaking up the process into three blocks was an effective way to design the overall application. Currently, a user can input a recording of a device playing white noise and the program will algorithmically derive a filter. Then, standard compression and distortion coefficients will be created for the device, to be adjusted based on user feedback. This streamlined process worked well during the troubleshooting phase, as the source of any arising issues would become more easily identifiable. In addition, many of the device models had similar sonic features to the actual device speakers. That being said, there is room for improvement, as well as expansion, in the real-time simulation. Most notably, the buffer by buffer processing lacks any sort of overlap, making the output sound choppy. A future task would involve finding a way to overlap-add to allow for cleaner processing. Adding more devices to the library would also make the application more well-rounded and applicable in a real-world setting.

7. REFERENCES

[1] Gamry Instruments, Inc. (2018, December 20). *Total Harmonic Distortion: Theory and Practice*. Electrochemical Instruments-Galvanostat/Potentiostat Manufacturer. https://www.gamry.com/application-notes/EIS/total-harmonic-distortion/